DIY Parallel Data Analysis

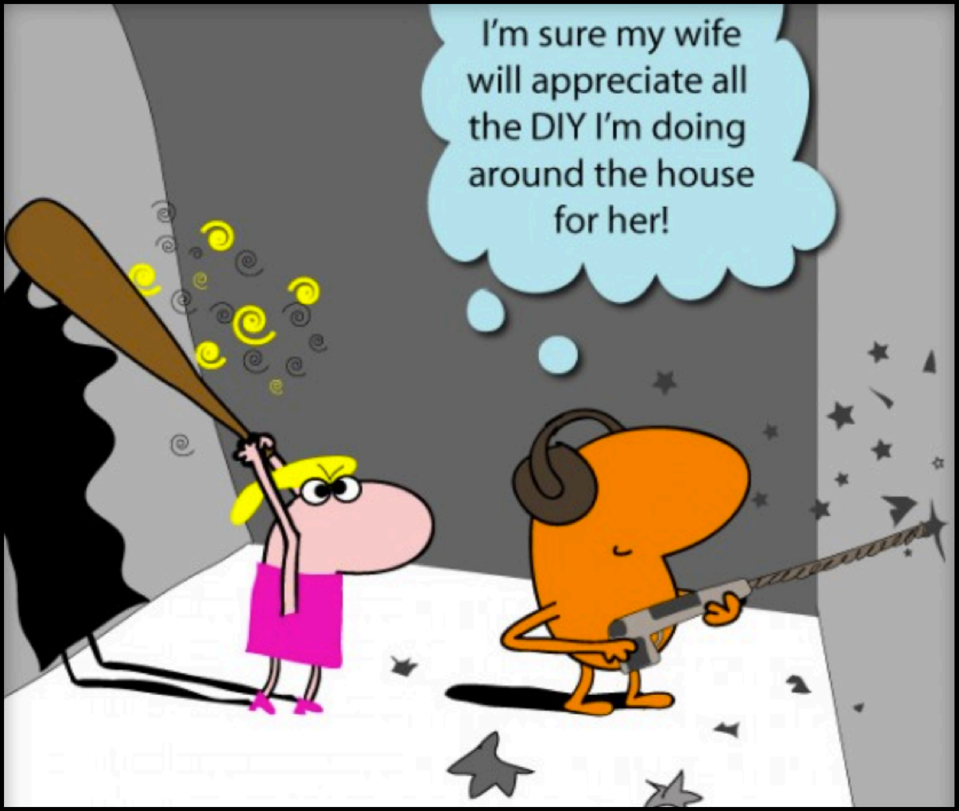


Image courtesy pigtimes.com

Tom Peterka

tpeterka@mcs.anl.gov

Mathematics and Computer Science Division

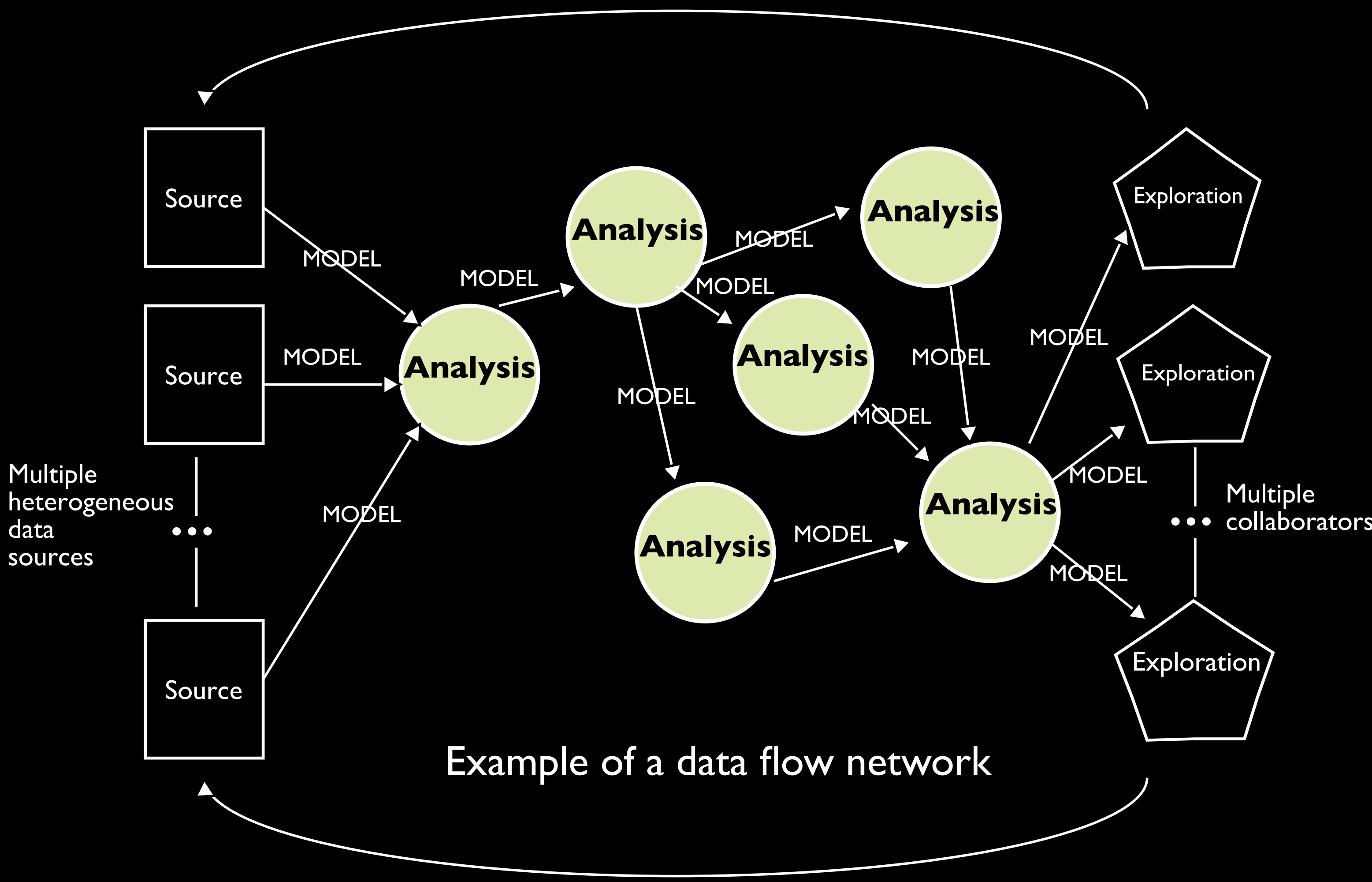
Dagstuhl Talk 6/17/13

Scientific Data Analysis Today

- Big science = big data, and
  - Big data analysis => big science resources
- Data analysis is data intensive.
  - Data intensity = data movement.
- Parallel = data parallel (for us)
  - Big data => data decomposition
  - Task parallelism, thread parallelism, while important, are not part of this work
- Most analysis algorithms are not up to the challenge
  - Either serial, or
  - Communication and I/O are scalability killers

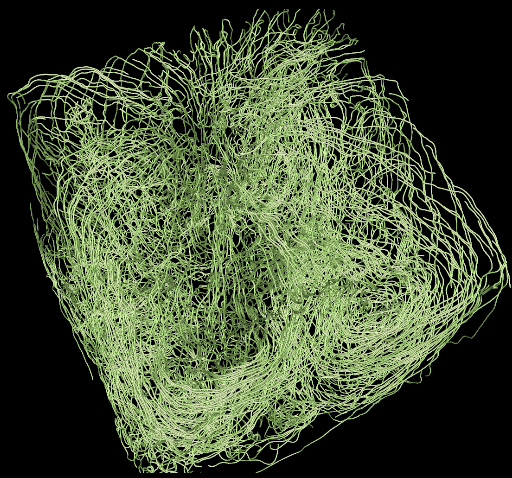
Definition of Data Analysis

- Any data transformation, or a network or transformations.
- Anything done to original data beyond its original generation.
- Can be visual, analytical, statistical, or data management.



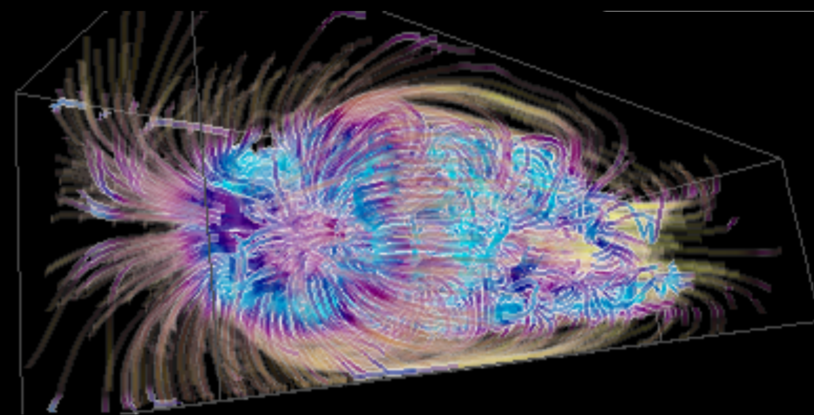
Example of a data flow network

Data Analysis Comes in Many Flavors



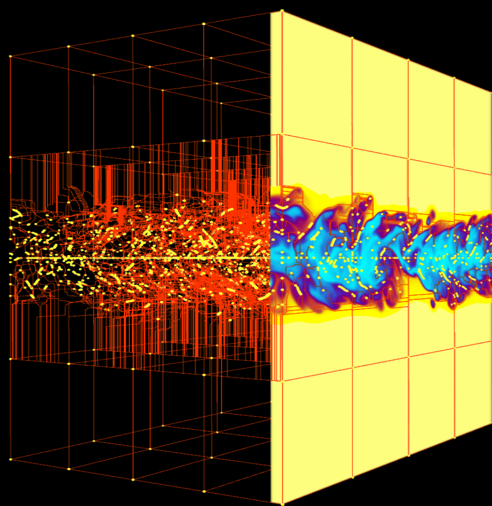
Visual

Particle tracing of thermal hydraulics flow



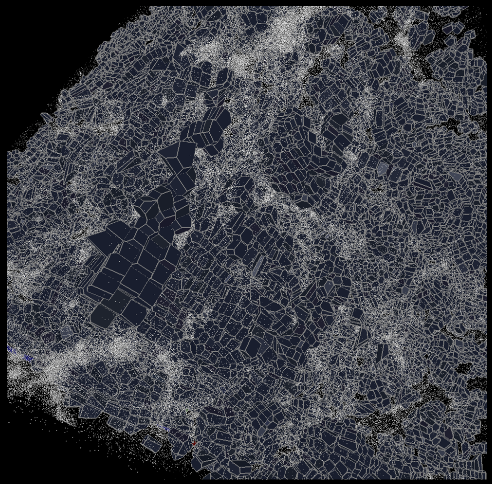
Statistical

Information entropy analysis of astrophysics



Topological

Morse-Smale Complex of combustion



Geometric

Voronoi tessellation of cosmology

You Have Two Choices to Parallelize Data Analysis

By hand	or	With tools

By hand

Application

Analysis Algorithm

Stochastic	Linear Algebra	Iterative	Nearest Neighbor

OS / Runtime

```
void ParallelAlgorithm() {
  ...
  MPI_Send();
  ...
  MPI_Recv();
  ...
  MPI_Barrier();
  ...
  MPI_File_write();
}
```

Interface

With tools

Application

Analysis Algorithm

Stochastic	Linear Algebra	Iterative	Nearest Neighbor

Data Movement

OS / Runtime

```
void ParallelAlgorithm() {
  ...
  LocalAlgorithm();
  ...
  DIY_Merge_blocks();
  ...
  DIY_File_write()
}
```

DIY

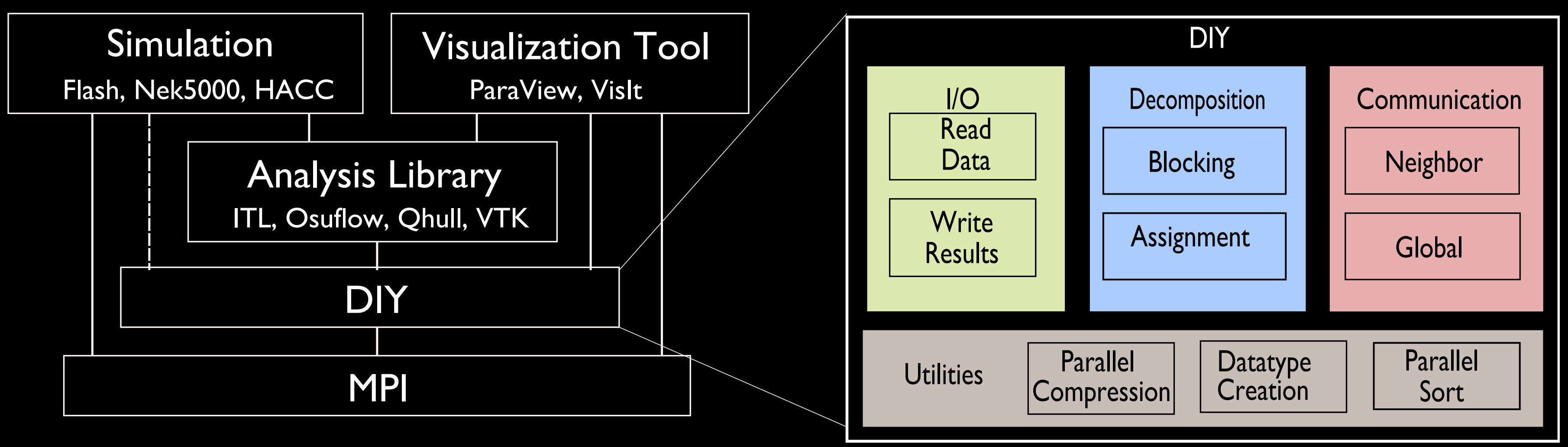
helps the user write data-parallel analysis algorithms by decomposing a problem into blocks and communicating items between blocks.

Features

- Parallel I/O to/from storage
- Domain decomposition
- Network communication
- Utilities

Library

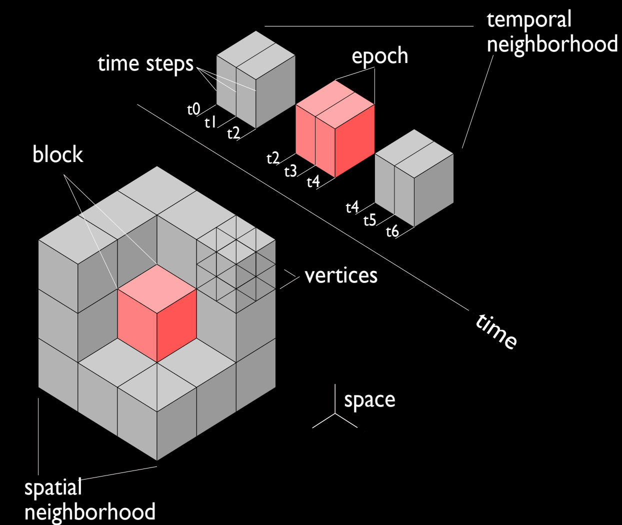
- Written in C++ with C bindings
- Autoconf build system (configure, make, make install)
- Lightweight: libdiy.a 800KB
- Maintainable: ~15K lines of code, including examples

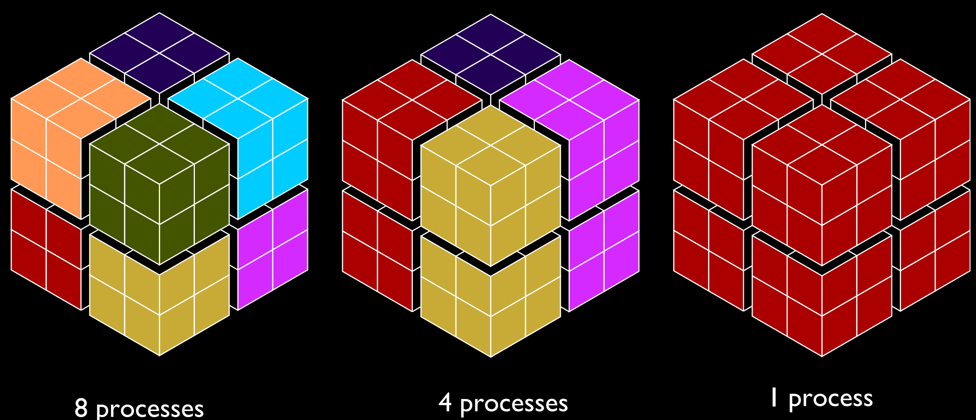


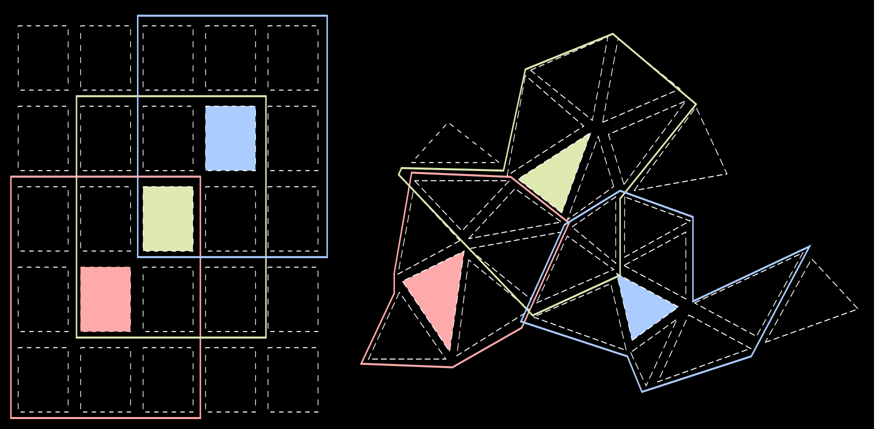
DIY usage and library organization

Nine Things That DIY Does

1. Separate analysis ops from data ops
2. Group data items into blocks
3. Assign blocks to processes
4. Group blocks into neighborhoods
5. Support multiple multiple instances of 2, 3, and 4
6. Handle time
7. Communicate between blocks in various ways
8. Read data and write results
9. Integrate with other libraries and tools







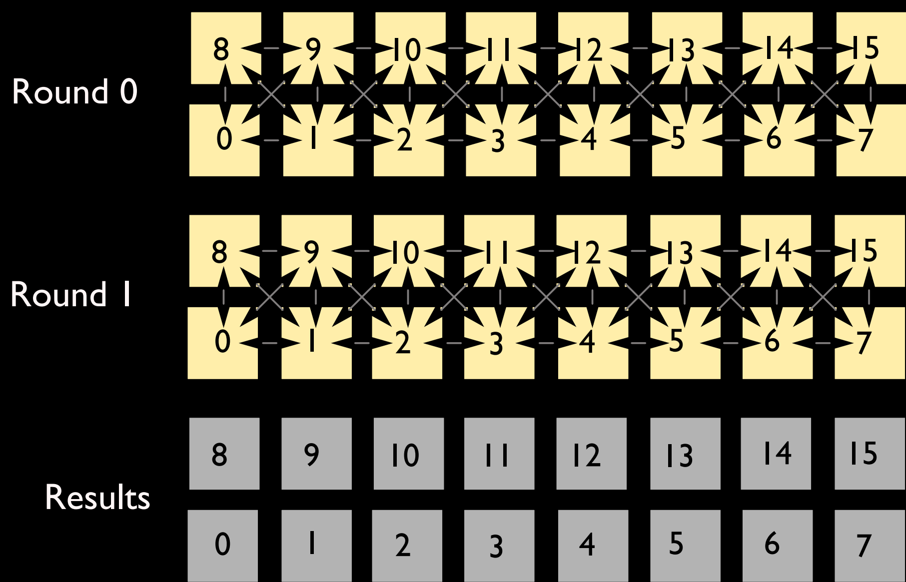
Two examples of 3 out of a total of 25 neighborhoods

Data Movement Patterns

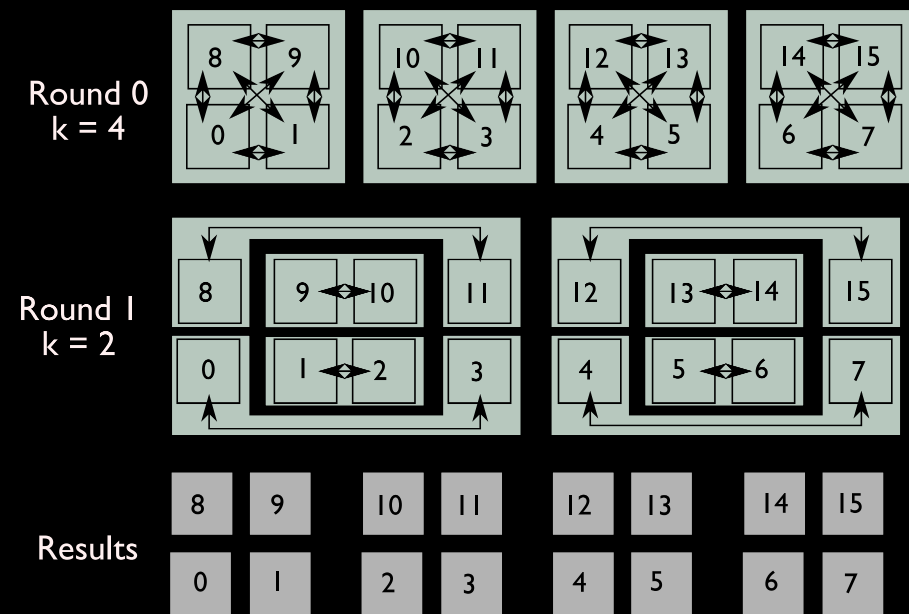
	Analysis	Communication	
Regular	Sort-Last Rendering	Swap-Based Reduction	Homogeneous Data
	Morse-Smale Complex	Merge-Based Reduction	Heterogeneous Data
	Information Entropy	Merge-Based Reduction	
Semi Regular	Particle Tracing	Neighborhood Exchange	
	Voronoi Tessellation	Neighborhood Exchange	
Irregular	Graph layout	Send-Receive	

Many different analysis operations share a small set of communication patterns. These communication kernels together with supporting utilities for decomposition and I/O can be encapsulated, optimized, and reused.

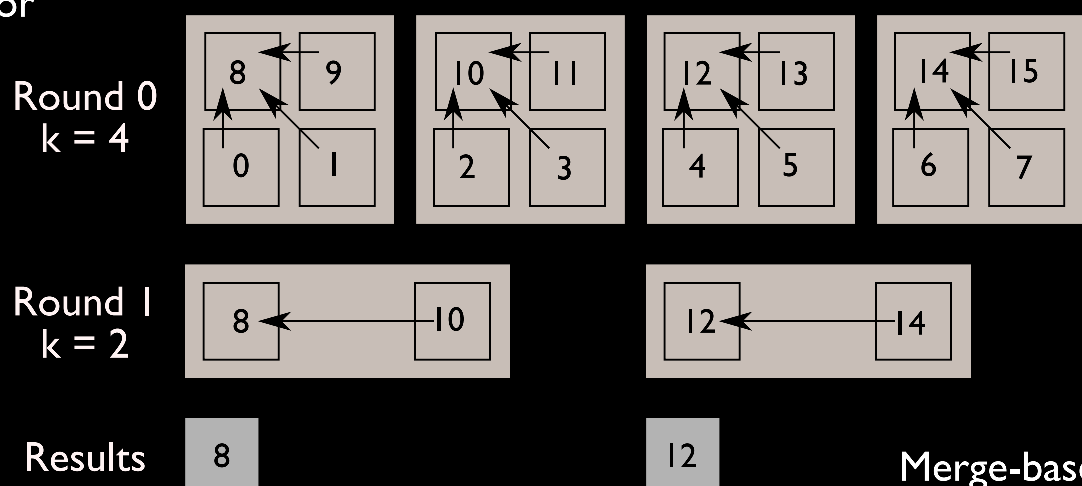
3 Communication Patterns



Nearest neighbor



Swap-based reduction



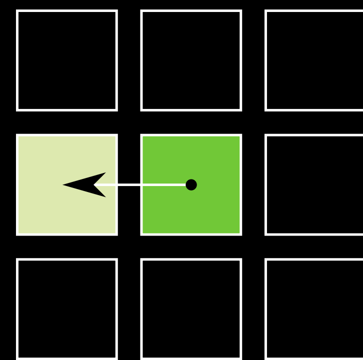
Merge-based reduction

Different Neighborhood Communication Patterns

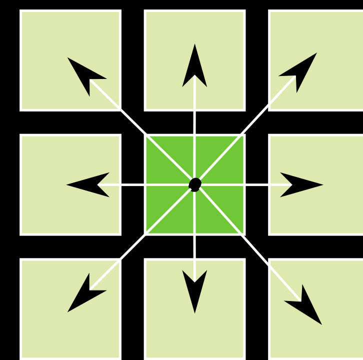
DIY provides point to point and different varieties of collectives within a neighborhood via its enqueue_item mechanism. Items are enqueued are subsequently exchanged (2 steps).

How to enqueue items for neighbor exchange

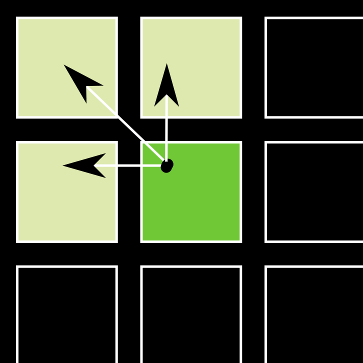
- DIY offers several options
- Send to a particular neighbor or neighbors, send to all nearby neighbors, send to all neighbors
- Support for periodic boundary conditions involves tagging which neighbors are periodic and calling user-defined transform on objects being sent to them



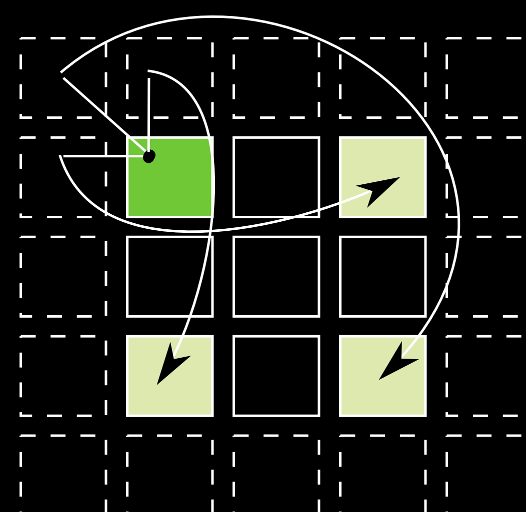
Send to only specific neighbors, indicated in various ways



Send to all neighbors



Send to all neighbors near enough to a target point

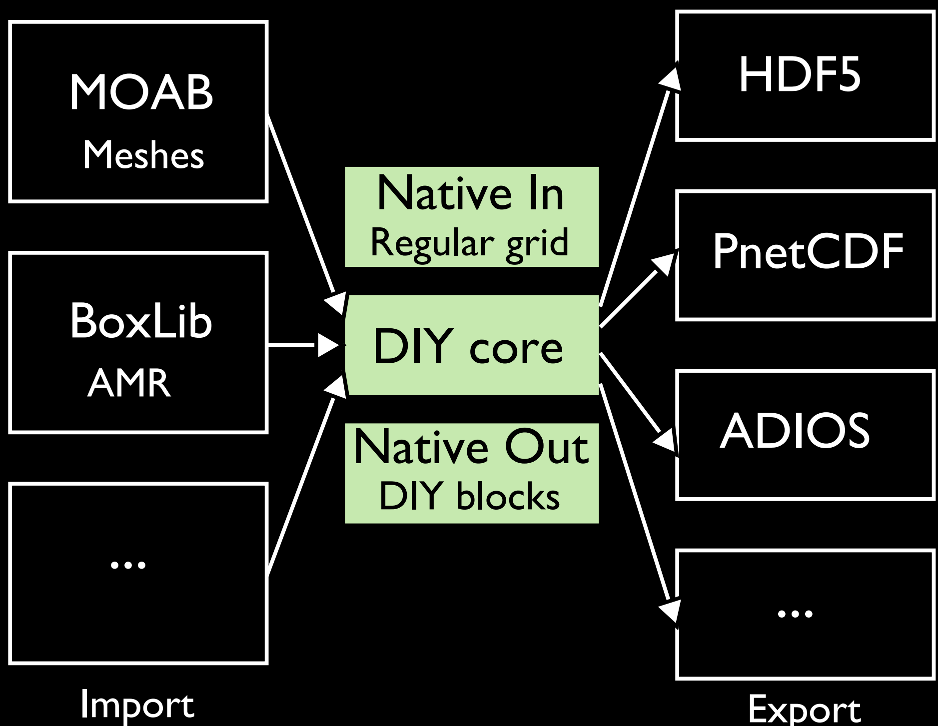


Support for wraparound neighbors (periodic boundary conditions)

Interaction with Other Libraries

DIY by design doesn't include input or output data models. Rather than re-inventing them, it can import and export those models.

Import: Replicate model using DIY_Decomposed(), explicitly providing blocks and neighbors to DIY
Export: Just use the other model API. DIY does not prevent you from making other library calls.



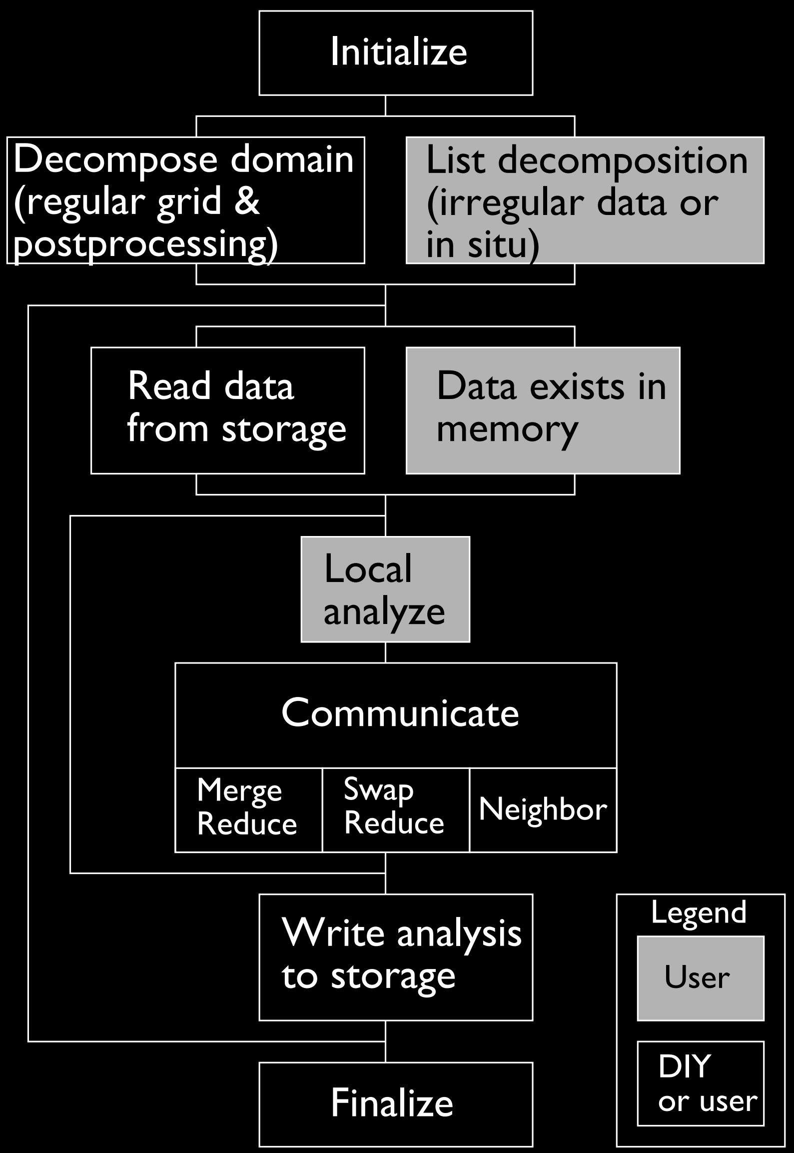
Writing a DIY Program

Documentation

- README for installation
- User's manual with description, examples of custom datatypes, complete API reference

Tutorial Examples

- Block I/O: Reading data, writing analysis results
- Static: Merge-based, Swap-based reduction, Neighborhood exchange
- Time-varying: Neighborhood exchange
- Spare thread: Simulation and analysis overlap
- MOAB: Unstructured mesh data model
- VTK: Integrating DIY communication with VTK filters
- R: Integrating DIY communication with R stats algorithms
- Multimodel: multiple domains and communicating between them



Published Performance and Scalability

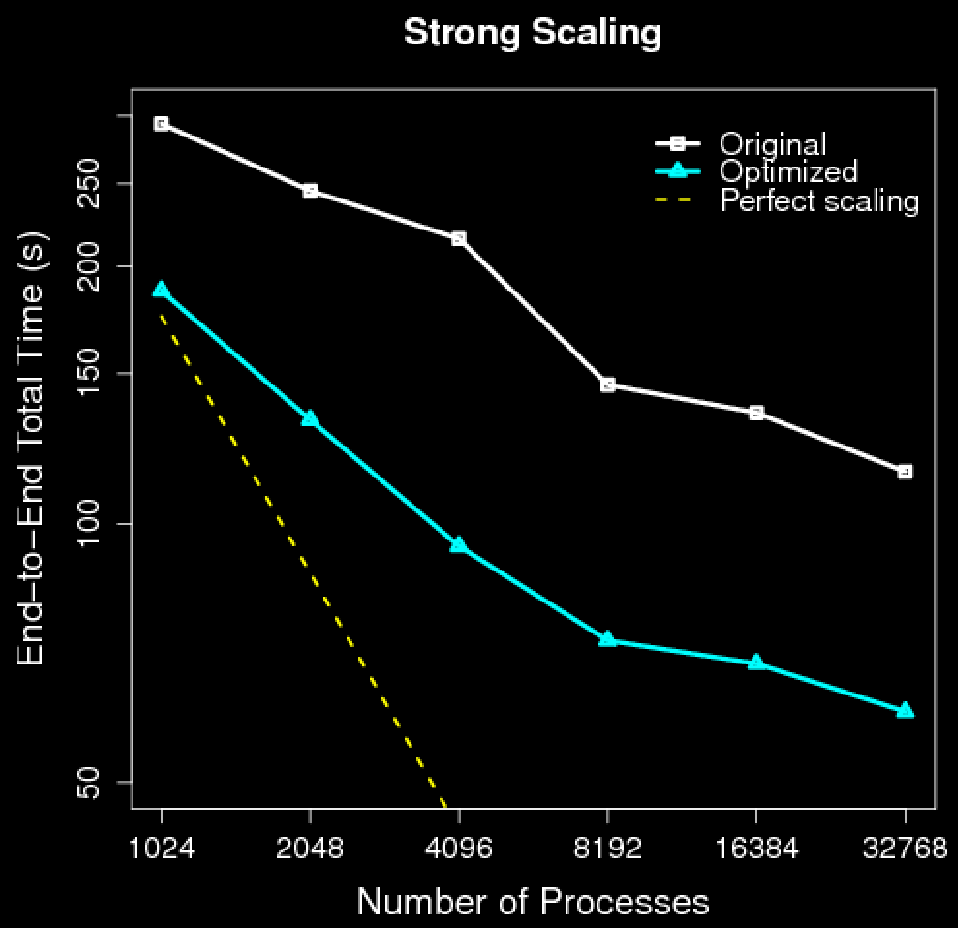
DIY

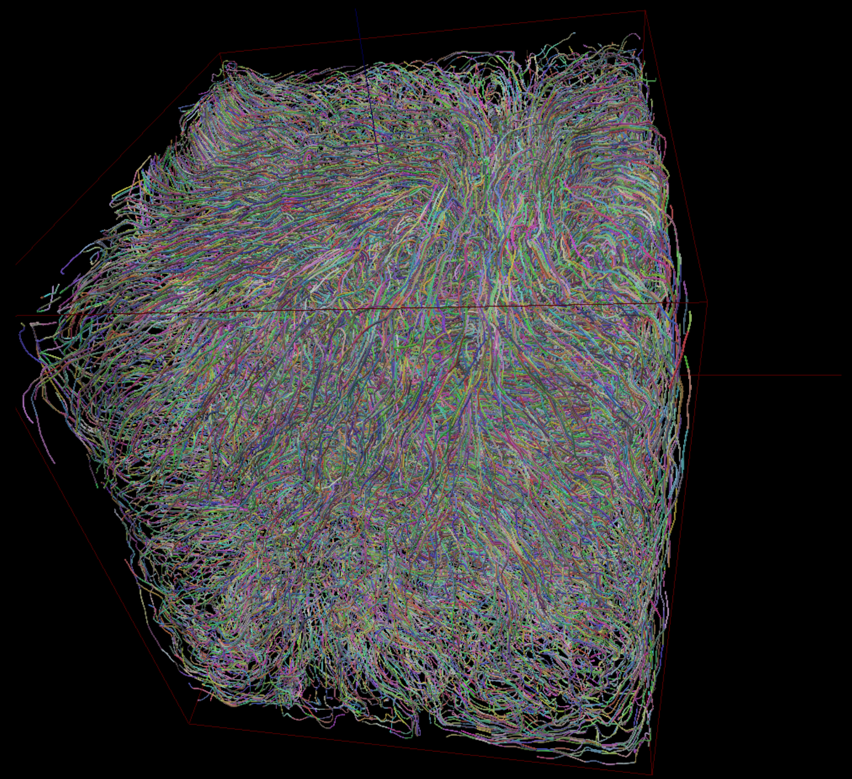
- Peterka, T., Ross, R., Kendall, W., Gyulassy, A., Pascucci, V., Shen, H.-W., Lee, T.-Y., Chaudhuri, A.: Scalable Parallel Building Blocks for Custom Data Analysis. Proceedings of Large Data Analysis and Visualization Symposium (LDAV'11), IEEE Visualization Conference, Providence RI, 2011.
- Peterka, T., Ross, R.: Versatile Communication Algorithms for Data Analysis. 2012 EuroMPI Special Session on Improving MPI User and Developer Interaction IMUDI'12, Vienna, AT.

DIY applications

- Peterka, T., Ross, R., Nouanesengsey, B., Lee, T.-Y., Shen, H.-W., Kendall, W., Huang, J.: A Study of Parallel Particle Tracing for Steady-State and Time-Varying Flow Fields. Proceedings IPDPS'11, Anchorage AK, May 2011.
- Gyulassy, A., Peterka, T., Pascucci, V., Ross, R.: The Parallel Computation of Morse-Smale Complexes. Proceedings of IPDPS'12, Shanghai, China, 2012.
- Nouanesengsy, B., Lee, T.-Y., Lu, K., Shen, H.-W., Peterka, T.: Parallel Particle Advection and FTLE Computation for Time-Varying Flow Fields. Proeedings of SC12, Salt Lake, UT.
- Peterka, T., Kwan, J., Pope, A., Finkel, H., Heitmann, K., Habib, S., Wang, J., Zagaris, G.: Meshing the Universe: Integrating Analysis in Cosmological Simulations. Proceedings of the SC12 Ultrascale Visualization Workshop, Salt Lake City, UT.
- Chaudhuri, A., Lee-T.-Y., Zhou, B., Wang, C., Xu, T., Shen, H.-W., Peterka, T., Chiang, Y.-J.: Scalable Computation of Distributions from Large Scale Data Sets. Proceedings of 2012 Symposium on Large Data Analysis and Visualization, LDAV'12, Seattle, WA.

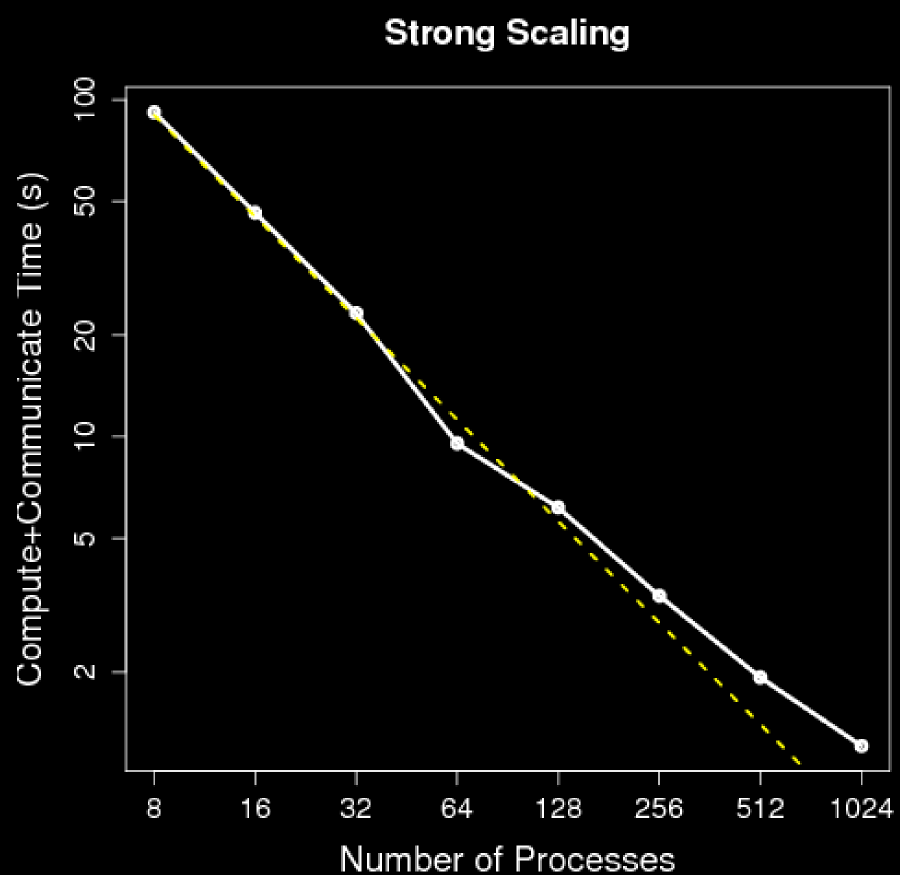
Particle Tracing

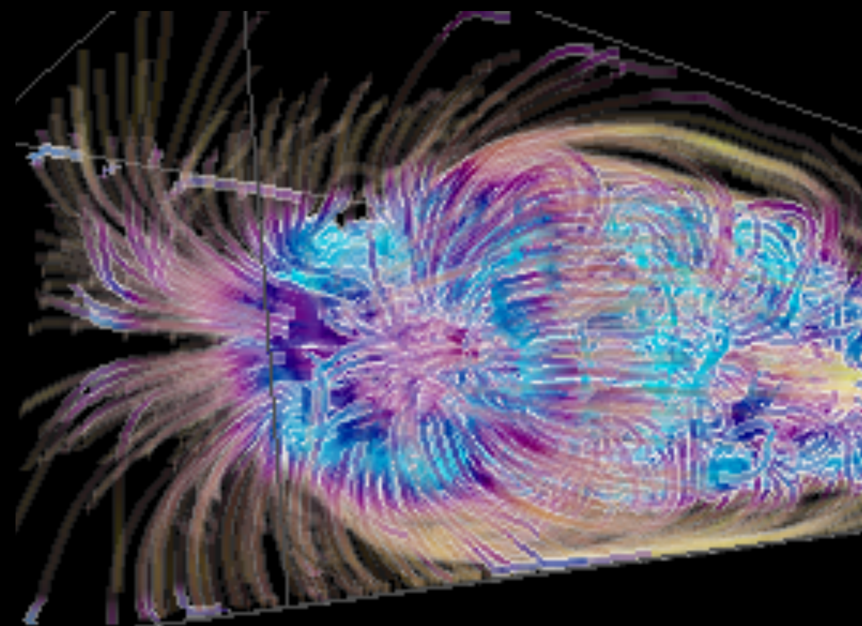




Particle tracing of ¼ million particles in a 2048^3 thermal hydraulics dataset results in strong scaling to 32K processes and an overall improvement of 2X over earlier algorithms

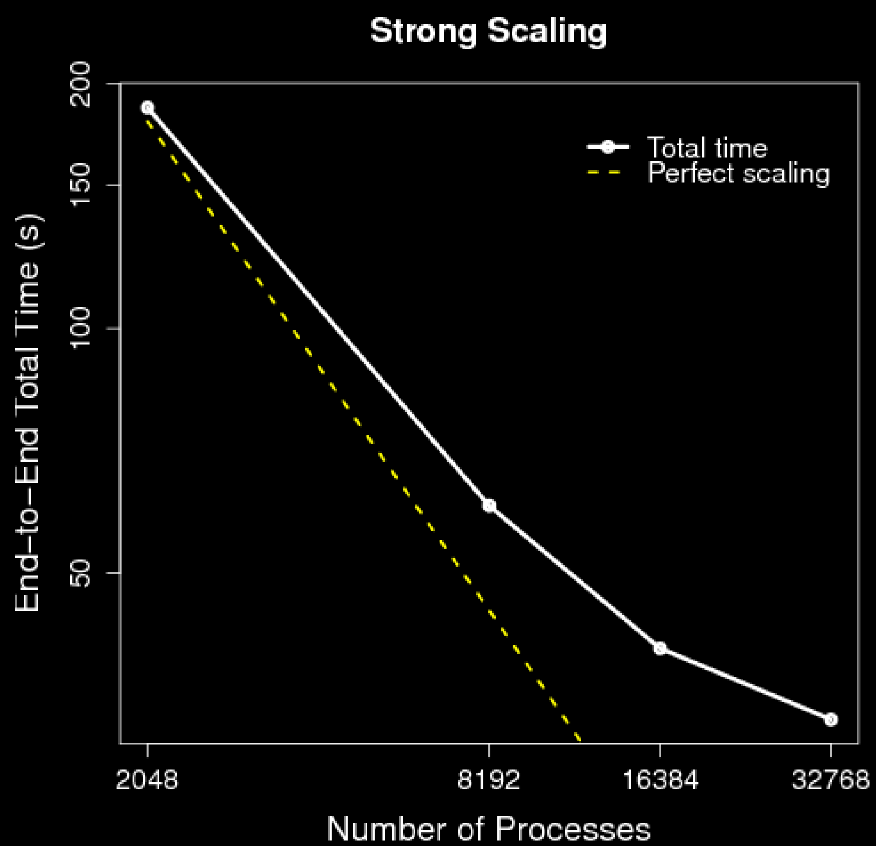
Information Entropy

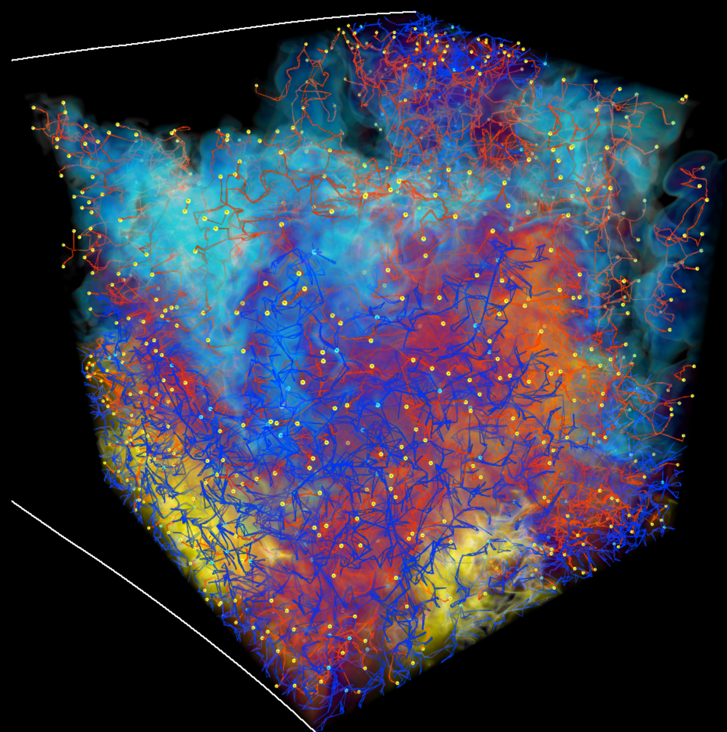




Computation of information entropy in 126x126x512 solar plume dataset shows 59% strong scaling efficiency.

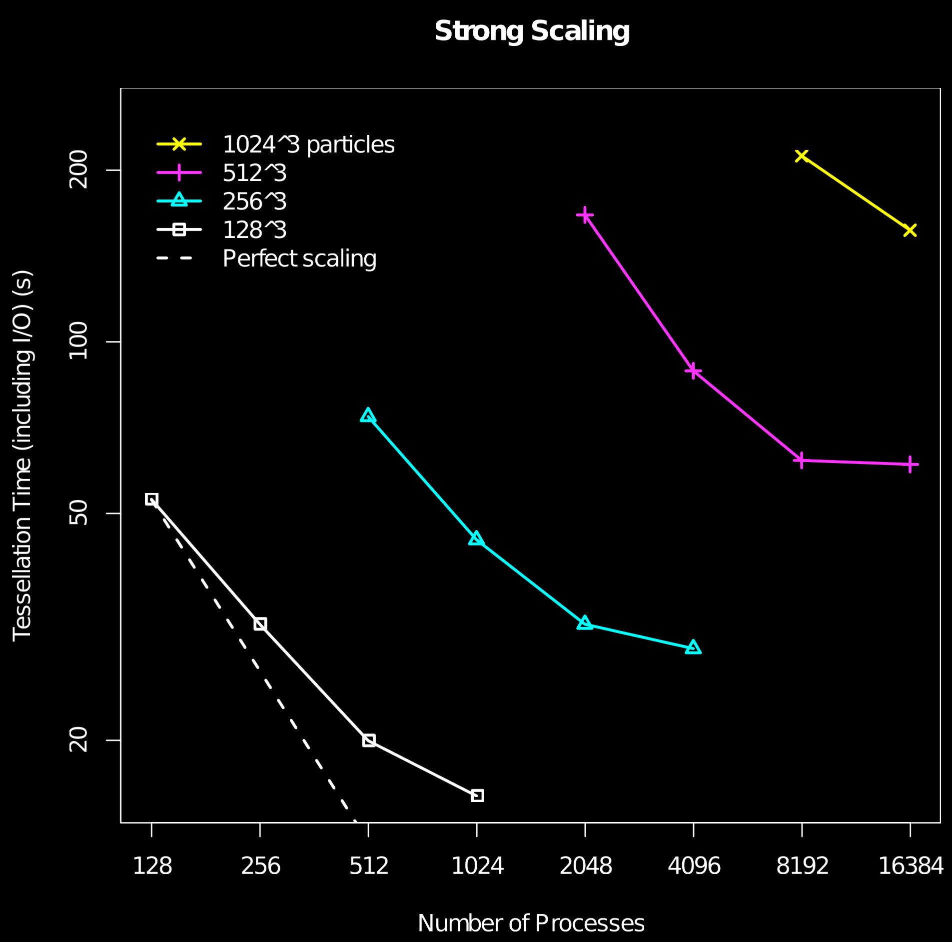
Morse-Smale Complex

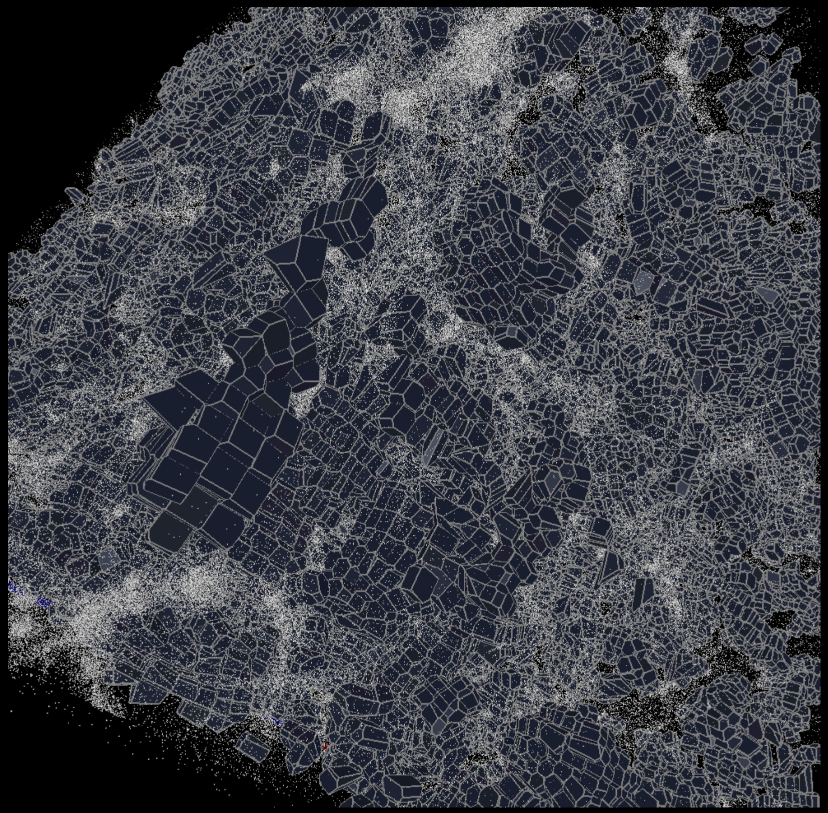




Computation of Morse-Smale complex in 1152^3 Rayleigh-Taylor instability data set results in 35% end-to-end strong scaling efficiency, including I/O.

In Situ Voronoi Tessellation





For 128^3 particles, 41 % strong scaling for total tessellation time, including I/O; comparable to simulation strong scaling.

Recap and Looking Ahead

Done: Benefits

- Productivity
  - Express complex algorithms flexibly
    - Multiple blocks per process
    - Complete / partial reductions
    - Neighbor inclusion and communication
  - Simplify existing tasks
    - Custom data type creation
    - Compression
- Performance
  - Published scalability
  - Configurable algorithms

To Do: Research Directions

- Advanced decomposition
  - Block groups
- Improved communication algorithms
  - Less synchronous, more overlap with computation
- High-level communication operations
  - Ghost cell exchange, kernel convolution (stencil)
- Load balancing
  - Block overloading, dynamic reassignment
- Programming models
  - MPI + X on Mira, Titan
- Usability
  - Improved API

Acknowledgments:

Facilities
Argonne Leadership Computing Facility (ALCF)
Oak Ridge National Center for Computational Sciences (NCCS)

Funding
DOE SDMAV Exascale Initiative
DOE Exascale Codesign Center
DOE SciDAC SDAV Institute

http://www.mcs.anl.gov/~tpeterka/software.html

https://svn.mcs.anl.gov/repos/diy/trunk

Tom Peterka
tpeterka@mcs.anl.gov
Mathematics and Computer Science Division